CS320 Summary and Reflections Report

Trevor Mickelson

Southern New Hampshire University

Table of Contents

[Table of Contents 2](#_Toc121853840)

[1. Summary 3](#_Toc121853841)

[1a. Describe your unit testing approach for each of the three features. 3](#_Toc121853842)

[1b. Describe your experience writing the JUnit tests. 4](#_Toc121853843)

[2. Reflection 4](#_Toc121853844)

[2a. Testing Techniques 4](#_Toc121853845)

[2b. Mindset 5](#_Toc121853846)

[References 6](#_Toc121853847)

CS320 Summary and Reflections Report

# 1. Summary

## 1a. Describe your unit testing approach for each of the three features.

### Contact Entity and Contact Service.

#### To what extent was your approach aligned with the software requirements?

My approach to all three features was the same. I wanted to create tests for every inch of the software, but not in such a way that was hard coded to only the software that was currently being asked of me. I wanted the software to be scalable in a way that made sense, and I wanted to avoid the fragile test problem. Which at smaller scales is arbitrary, but as the project scales up, would become a crutch. I achieved goals by testing a shared function once, rather than each time it was called.

requirements. Support your claims with specific examples and evidence.]

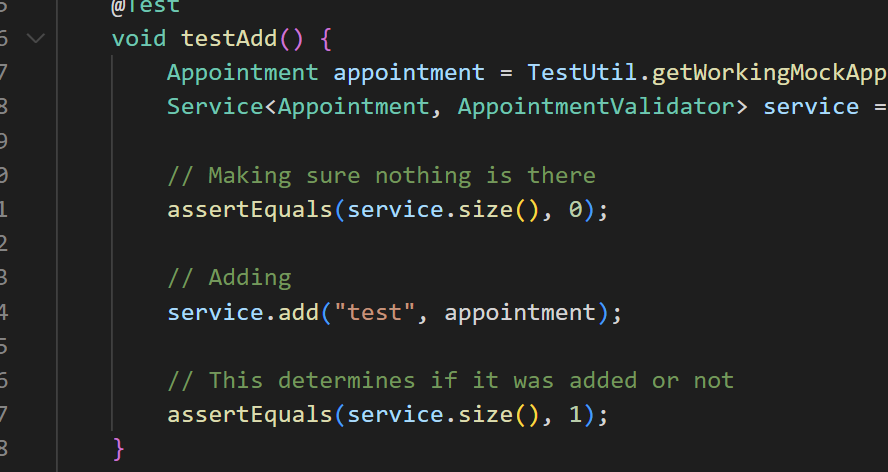
#### Defend the quality of your JUnit tests.

### 

In this specific example, I’m checking to see if the function can do exactly what it says. Can it check if the string is not null, and if it’s within a specified length? I start by testing for null first, then I test for an invalid string length, then I test for 2 correct inputs, and lastly, I check for a correct string, but an invalid length. These tests showed me this function worked completely.

In terms of the services system, I’m quite confident that this works completely, and meets all of the software and testing requirements. I simply created a single services class that each system could hook into, which allowed me to only need to test 1 instance of the services system. Here’s some examples of what I did there.

Text

Description automatically generated

### Task Entity and Task Service.

#### To what extent was your approach aligned with the software requirements?

My approach to the Task Entity and Task Service was the same as everything else, in fact nothing really changed at all. As I created a universal testing system that tested all classes, rather than each one individually.

#### Defend the quality of your JUnit tests.

I’ve proven the quality of my JUNIT tests for this in the example above.

### Appointment Entity and Appointment Service.

#### To what extent was your approach aligned with the software requirements?

[Explain how your approach to testing the Appointment Service was aligned with the software requirements. Support your claims with specific examples and evidence.]

#### Defend the quality of your JUnit tests.

Because the way I handled my tests was one for all, I’ve answered this question up above.

## 1b. Describe your experience writing the JUnit tests.

### How did you ensure that your code was technically sound?

The software testing techniques I used were all based around testing edge cases, or just testing if the function itself works. This allowed me to ensure the code was technically sound, rather than just guessing and hoping it was good.

### How did you ensure that your test code was efficient?

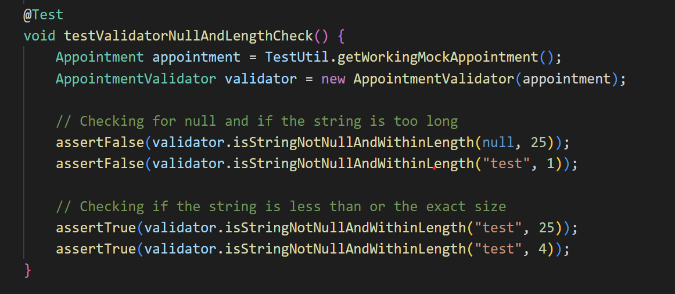
* In terms of efficiency, I suppose I never technically tested for that. I did make sure all my code ran at a time complexity of 0(1) wherever I could, but I never actually tested the amount of time it took to run the functions. That’s an interesting idea, one I had not thought of.

# 2. Reflection

## 2a. Testing Techniques

### What were the software testing techniques that you employed in this project?

The testing techniques I used in this project were to make the code re-usable and make sure I take every edge case into consideration. A specific example is one I’ve shown already, but I’ll show it again for the sake of argument.



### What are the other software testing techniques that you did not use for this project?

I suppose one technique that wasn’t used is manual testing, or just testing in an actual work environment. The environment here was just a testing environment inside of an IDE. The reason for not using manual testing is just because it wasn’t the assignment, and it’s not logical here anyways. Well, it could be faster maybe with tiny functions but in the case of unit testing, at least I’m 100% confident my functions work.

### For each technique you discussed, explain their practical uses and implications for different software projects and situations.

I believe there’s practical applications to both techniques I used. It’s very important to write re-usable code rather than hard coding everything. And it’s also applicable in basically every programming application to write tests for functions that check for edge cases. I believe this type of thing is probably more useful on larger projects, where on smaller projects manual testing could be faster.

## 2b. Mindset

### Assess the mindset you adopted working on this project.

The biggest mindset shift for me is writing literal tests themselves, whereas I’m more adjusted to manually testing everything.

### Assess the ways you tried to limit bias in your review of the code.

It’s very hard to remove bias from code testing, and to be honest I don’t think it’s completely possible. When you write something, you understand it fully, but for whatever reason this makes it harder to break it. I’ve found that external testers will always try something different than the creator does, and often times break something with some unknown edge case. Unfortunately, this only really works with manual testing.

### Finally, evaluate the importance of being disciplined in your commitment to quality as a software engineering professional.

* Being disciplined when programming is not just related to tests. It’s extremely important to go well and not too fast. Going fast always results in something overlooked, or extremely coupled code that can’t scale easily.

References

Last Name, F. M. (Year). Article Title. *Journal Title*, Pages From - To.

Last Name, F. M. (Year). *Book Title.* City Name: Publisher Name.